- Requirements analysis allows the software engineer (sometimes called *analyst* in this role) to refine the software allocation and build models of the data, functional, and behavioural domains that will be treated by software.
- Software requirements analysis may be divided into five areas of effort:
    1. Problem recognition
    2. Evaluation and synthesis
    3. Modelling
    4. Specification
    5. Review.
- Initially, the analyst studies the **System Specification** (if one exists) and the **Software Project Plan** inorder to understand software in a system context and to review the software scope.
- Next, communication for analysis must be established so that problem recognition is ensured. The goal is **recognition** of the basic problem elements as perceived by the customer/users.
- **Problem evaluation and solution synthesis** is the next major area of effort for analysis. The analyst must define all externally observable data objects, evaluate the flow and content of information, define and elaborate all software functions, understand software behaviour in the context of events that affect the system, establish system interface characteristics, and uncover additional design constraints. Each of these tasks serves to describe the problem so that an overall approach or solution may be synthesized.
- Throughout evaluation and solution synthesis, the analyst's primary focus is on "what," not "how." What data does the system produce and consume, what functions must the system perform, what behaviours does the system exhibit, what interfaces are defined and what constraints apply?
- During the evaluation and solution synthesis activity, the analyst creates models of the system in an effort to better understand data and control flow, functional processing, operational behaviour, and information content. The model serves as a foundation for software design and as the basis for the creation of specifications for the software.

Requirement Gathering Techniques

Software requirements analysis always begins with communication between two or more parties. A customer has problem that may be open to to a computer-based solution. A developer responds to the customer’s request for help. The steps in analysis process are:

(1) Initiating the process
(2) Facilitated Application Specification Techniques (FAST)
(3) Quality Function Deployment (QFD)
(4) Use-Case Scenarios

1. **Initiating the process:-**
    - The most commonly used requirements gathering technique is to conduct a meeting or interview.
    - The first meeting between a software engineer (the analyst) and the customer is full of confusions.
    - Neither person knows what to say or ask; both are worried that what they do say will be misinterpreted; both are thinking about where it might lead (both likely have radically different expectations here); both want to get the thing over with, but at the same time, both want it to be a success.
    - The analyst should start by asking *“context free questions”*. That is, a set of questions that will lead to a basic understanding of the problem, the people who want a solution, the nature of the solution that is desired, and the effectiveness of the first encounter itself. The first set of context-free questions focuses on the customer, the overall goals, and the benefits.
    - For example, the analyst might ask: Who is behind the request for this work?
        - Who will use the solution?
        - What will be the economic benefit of a successful solution?
        - Is there another source for the solution that you need?
    - The next set of questions enables the analyst to gain a better understanding of the problem and the customer to voice his or her perceptions about a solution:
        - How would you characterize "good" output that would be generated by a successful solution?
        - What problem(s) will this solution address?
        - Can you show me (or describe) the environment in which the solution will be used?

   - Will special performance issues or constraints affect the way the solution is approached?
- The final set of questions known as "*Meta-questions*" focuses on the effectiveness of the meeting propose the following list:
   - Are you the right person to answer these questions? Are your answers "official"?
   - Are my questions relevant to the problem that you have?
   - Am I asking too many questions?
   - Can anyone else provide additional information?
   - Should I be asking you anything else?
- These questions will help to "break the ice" (to initiate a friendly conversation) and initiate the communication that is essential to successful analysis.

2. Facilitated Application Specification Techniques (FAST)(Collaborative Requirements Gathering):-

- FAST approach encourages the creation of a joint team of customers and developers, who work together to identify the problem, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements.
- This approach is similar to brainstorming session. The objective of the approach is to bridge the expectation gap – a difference between what developers think they are supposed to build and what customers think they are going to get.
- FAST is predominantly used. Basic guidelines for approaching FAST are as follows:
   - A meeting is conducted at a neutral site and attended by both developers and customers.
   - Rules for preparation and participation are established.
   - An agenda is suggested that is formal enough to cover all-important points but informal enough to encourage the free flow of ideas.
   - A "facilitator" (who can be a customer, a developer or an outsider) controls the meeting.
   - A "definition mechanism"(which can be work sheets, flip charts, or a wall board) is used.
   - The goal is to identify the problems, propose elements of the solution, negotiate different approaches, and specify a preliminary set of solution requirements in an atmosphere that is conducive to the accomplishment of the goal.
- To better understand the flow of events in a typical FAST meeting, a brief scenario that outlines the sequence of events that lead up to the meeting, occurs during the meeting, and follows the meeting.
- As the FAST meeting begins, the first topic of discussion is the need and justification for the new product – everyone should agree that the product justified.
- Once agreement has been established, each participant presents his or her list of objectives and services required, and these lists are displayed in the meeting.
- Discussions are held on these objectives and services. And after this, combined lists for these topics are prepared by eliminating redundant entries and adding new ideas.
- The combined list is shortened, lengthened, or reworded to properly reflect the product or system to be developed. The objective is to develop a consensus list in each topic area.
- And the same process is repeated many times and many sessions are held for this; till a consensus (agreement) is reached.
- The FAST team is composed of representatives from marketing, software and hardware engineering, and manufacturing.
- The team approach of FAST provides the benefits of many points of view, instantaneous discussion and refinement, and is a concrete step toward the development of a specification.

3. Quality Function Deployment:-

- Quality function deployment is a quality management technique that translates the needs of the customer into technical requirements for the software.
- It concentrates on maximizing the customer satisfaction. To accomplish this, QFD emphasizes understanding of what is valuable to the customer and then deploying these values throughout the engineering process.

QFD identifies three types of requirements:

1. **Normal requirements.** Objectives and goals are stated for a product or system during meetings with the customer. If these requirements are present, the customer is satisfied. Examples of normal requirements might be requested types of graphical displays, specific system functions, and defined levels of performance.

2. **Expected requirements.** These requirements are implicit to the product or system and may be so fundamental that the customer does not explicitly state them. Their absence will be cause for significant dissatisfaction. Examples of expected requirements are: ease of human/machine interaction, overall operational correctness and reliability, and ease of software installation.
3. **Exciting requirements.** These features go beyond the customer's expectations and prove to be very satisfying when present. For example, word processing software is requested with standard features. The delivered product contains a number of page layout capabilities that are quite pleasing and unexpected.

- In actuality, QFD spans the entire engineering process.
- In meetings with the customer, **function deployment** is used to determine the value of each function that is required for the system.
- **Information deployment** identifies both the data objects and events that the system must consume and produce. These are tied to the functions.
- Finally, **task deployment** examines the behaviour of the system or product within the context of its environment.
- **Value analysis** is conducted to determine the relative priority of requirements determined during each of the three deployments.

4. Use-Cases:-

- As requirements are gathered as part of informal meetings, FAST, or QFD, the software engineer (analyst) can create a set of scenarios that identify a thread of usage for the system to be constructed. The scenarios, often called ***use-cases***, provide a description of how the system will be used.
- To create a use-case, the analyst must first identify the different types of people (or devices) that use the system or product. These actors actually represent roles that people (or devices) play as the system operates. Defined somewhat more formally, an **actor** is anything that communicates with the system or product and that is external to the system itself.
- Once actors have been identified, use-cases can be developed. The use-case describes the manner in which an actor interacts with the system.
- The following are the suggested questions that should be answered by the use-case:
    - What main tasks or functions are performed by the actor?
    - What system information will the actor acquire, produce, or change?
    - Will the actor have to inform the system about changes in the external environment?
    - What information does the actor desire from the system?
    - Does the actor wish to be informed about unexpected changes?
- In general, a use-case is simply a written narrative that describes the role of an actor as interaction with the system occurs.

Analysis Principles

Analysis should be based on a set of operational principles:

1. The information domain of a problem must be represented and understood.
2. The functions that the software is to perform must be defined.
3. The behaviour of the software (as a consequence of external events) must be represented.
4. The models that depict information function and behaviour must be partitioned in a manner that uncovers detail in a layered (or hierarchical) function.
5. The analysis process should move from essential information toward implementation detail.

In addition to the operational principles, a set of guiding principles for requirements engineering has been proposed as follows:

- Understand the problem before you begin to create the analysis model.
- Develop prototypes that enable a user to understand how human-machine interaction will occur.
- Record the origin of and the reason for every requirement.